SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V13I4P106

Volume 13 Issue 4, 81-93, April 2026

Original Article

Enhancing Ternary Content Addressable Memory Reliability with Error-Tolerant Convolutional Logic Code Synthesis

B. Suresh Kumar[1], Injeti Sowmya[2]

[1]*Department of Electrical, Electronics, and Communication Engineering, GITAM (Deemed to be University), Hyderabad, Telangana, India.*
[2]*Department of Electrical, Electronics, and Communication Engineering, GITAM (Deemed to be University), Visakhapatnam, Andhra Pradesh, India.*

[1]*Corresponding Author : sbolli@gitam.edu*

Received: 07 January 2026 Revised: 08 February 2026 Accepted: 07 March 2026 Published: 30 April 2026

Abstract *- In modern computing systems, memory reliability is crucial, as errors in storage and retrieval can lead to data corruption, system crashes, or performance degradation. Traditional RAMs employ error correction techniques such as Error-Correcting Code (ECC) memory and parity bits to detect and rectify bit flips. However, these methods suffer from limitations like increased latency, higher power consumption, and limited error correction capability, making them inadequate for high-speed, high-density memory applications. To overcome these drawbacks, this work proposes a TCAM Error Correction (TCAM-EC) approach, where data is stored within Ternary Content Addressable Memory (TCAM), which is susceptible to errors during read and write operations due to process variations, aging effects, and external disturbances. To ensure robust error detection and correction, the Error-Tolerant Convolutional Logic Code Synthesis (ET-CLCS) technique is introduced, leveraging convolutional encoding principles to efficiently identify and correct errors without significantly impacting performance or power efficiency. This method enhances memory reliability while maintaining the high-speed search and retrieval capabilities of TCAM, making it suitable for applications requiring fault-tolerant memory operations in data-intensive environments.*

Keywords *- Data Reliability, ECC Memory, Fault-Tolerant Operations, Memory Aging Effects, Read-Write Errors.*

1. Introduction

The development of the VLSI industry is still on a steady course, mainly because of the growing demand for high-performance and low-power-memory devices [1]. The advanced technologies, which include AI, big data, and IoT, require memory structures that can accommodate a large amount of data as per the usage requirement and provide low latency and data persistence. Thus, the Spin-Transfer-Torque Magnetoresistive Random-Access Memory (STT-MRAM) has recently received significant attention from both research and development, fulfilling important requirements for critical applications, such as high endurance, non-volatility, and low latency in application-specific settings, mainly for the edge of the networks.

Memory technologies are essential across industries and software services, including data centers, cloud, and fog, as well as AI applications [2]. Data centers are active networking environments, and an efficient high-bandwidth memory system is crucial for storing and processing huge volumes of data that pass through the centers. In edge computing applications, there is a need to have memory that is capable of computation in a real-time manner with shorter power consumption to enhance real-time learning in models of AI, since STT-MRAM has high reliability and change rate, which is significant for data reliability and requires sophisticated update frequency in the industrial IoT and automotive markets.

Dynamic Random-Access Memory (DRAM) and flash memory have been customary memory technologies that have supported computing systems for decades [3]. Still, these technologies pose challenges in establishing efficient performance within the energy needed by today's applications. Although DRAM has very low latency, it is volatile and consumes power, particularly when manufactured at small micron nodes [4]. However, Flash memory has disadvantages such as high cost, low interface performance, low endurance, and its non-volatile nature [5]. To overcome these challenges, some forms of error detection

and correction have been incorporated into memories. For instance, ECC memory [6] can identify and even eliminate single errors that have occurred, depending on the data reliability. However, when implementing ECC memory, it means extra cycle delay and power consumption due to overheads corresponding to redundant bits [7].

Additionally, ECC is mainly used against single-bit errors and is not sufficient to address multiple-bit errors or an error that occurred while doing the computation. A study was carried out by [8] that stresses that the need for stronger error detection and correction techniques is highly essential because modern memories are slowly shifting toward the In-Memory Computing (IMC) paradigm. Thus, the new and valuable findings of this work are as follows:

- To enhance memory reliability, the proposed TCAM-EC stores data in TCAM and applies adaptive error correction, overcoming the limitations of traditional RAM-based ECC.
- ET-CLCS is developed to detect and correct errors using Hybrid Convolutional Redundancy Encoding (HCRE), Parallelized Syndrome Computation (PSC), and Adaptive Convolutional Error Masking (ACEM) mechanisms.
- A PSC unit is introduced to improve error detection efficiency. This unit enables real-time multi-bit error localization with minimal latency.
- An ACEM scheme dynamically adjusts redundancy levels to optimize error correction, reducing power consumption and improving TCAM lifespan.

The rest of this paper is organized as follows: Section 2 gives a general overview of memory technologies and their error correction techniques. Section 3 explains the TCAM-EC method in detail, and Section 4 provides experimental results and performance evaluation. Finally, this study's conclusion is presented in section 5 before suggestions for future research are discussed.

2. Literature Survey

A scheme of safe in-memory computation with optimized ECC is proposed by Parrini et al. [9], which was named Error Detection and Correction (EDAC). Their implementation made it more reliable as the soft errors in volatile and non-volatile memory were addressed. The computation cost was, however, higher because of complicated parity check schemes. An in-situ error correction framework of multiple-bit computation in-memory accords was proposed by Lin et al. [10]. They maximized bit-level redundancy to boost in-memory architecture data integrity. The technique was afflicted with a large area overhead because of extra fault-tolerant structures. Wang et al. [11] introduced an Error-Tolerant CIM (ETCIM) processor, which provides error-free, redundancy-free repair and runtime Multiply-Accumulate (MAC) cell repair. This method removed the redundant repair circuits and maintained high accuracy. The main weakness was the more complex design in the case of the mitigation of the faults during runtime.

Jin et al. [12] created a code of a Latin Square Matrix (LSMC) in the correction of burst error in MRAM. Their methodology enhanced the reliability of memory by fixing large-scale bursts. However, the encoding and decoding latency was high and, thus, affected real-time applications. Bai et al. [13] suggested an SRAM Computer-In-Memory (SRAM-CIM) system with an enhanced compilation system, which included the weight mapping and error correction. This method enhanced the fault tolerance and was also energy efficient. The downside that was very major was the augmented memory access latency. Moon et al. [14] proposed a two-level ECC solution to High-Bandwidth Memory (HBM) over Soft Write Disturbance (SWD) errors. They used the method that took advantage of prediction-based correction to increase fault resilience. Nevertheless, there were storage needs that minimized the efficiency of the memory usage.

Gupta et al. [15] proposed a multicore system designed for STT-MRAM-based storage based on a cache memory equipped with highly effective error detection and correction measures. Their implementation enhanced the fault tolerance of cache architectures but created power consumption. Alhiyari et al. [16] suggested a software-defined failure recovery model of data center networks, which is TCAM-aware fault detection. Such a strategy enhanced the efficiency of recovery, yet the latency in identifying a failure had the negative effect of influencing the performance of the system.

Garzon et al. [17] introduced an approximate CAM that can perform searches and a tunable Hamming distance to provide better error correction, which is abbreviated as CAM-HD. Their design maximized the accuracy of associative searches and was able to tolerate small bit errors. Nevertheless, the method had a problem of scaling when using the large memory arrays.

Marani et al. [18] designed an IMC-based Hamming code implementation to correct errors. Their approach installed redundancy-mindful logic to enhance the resistance to errors in embedded systems. The weakness was the rise in energy consumption on deep submicron nodes. Yu et al. [19] suggested a three-dimensional (3D) self-rectifying memristive structure of TCAM logic implementation to perform voluminous in-memory search operations. Their system reduced bit errors and used the exact-match queries. Complexity in fabrication, however, restricted practical use. Sreekumar et al. [20] also incorporated error detection and correction technology into the RISC-V processor microarchitecture to increase the reliability of computations. They enhanced the fault tolerance of the processors, but at the cost of overhead silicon space.

Kang et al. [21] proposed fault-bounding on-die BCH codes to improve system ECC and minimize the chances of undetected errors. Their algorithm was low-latency error-correcting but lacked flexibility for various error shapes. Hemaram et al. [22] showed an adaptive error correction scheme of STT-MRAM, and to balance reliability with power efficiency, they applied asymmetric redundancy. Control, however, was limited by higher production costs. According to Tu et al. [23], raw bit error Rate-Conscious Polar Coding-Based 3D NAND flash memory (RaPC) was proposed, which seeks to optimize the ECC performance, depending on real-time error properties. Its main weakness was the enhanced computation complexity of the real-time adaptation.

Shin et al. [24] have come up with a reliability-optimized OD-ECC and S-ECC enhancement framework of HBM3 memory systems that enhances resilience to transient errors. This was constrained by the fact that multi-stage correction caused more processing latency. Jha et al. [25] proposed a Bayesian Analog Error-Mitigating Codes (BAEMC) scheme of robust in-memory computation by using probabilistic models to decrease fault tolerance. Nonetheless, the method demanded more computers, which influenced the effectiveness of the performance. Aziz et al. [26] gave a multi-bit error detection and correcting algorithm based on Horizontal-Vertical-Diagonal-Knight (HVDK) parity, which improves fault correction through redundancy. The higher parity overhead, however, affected data storage efficiency. Cano-Páez et al. [27] have suggested a Dynamic Partial Reconfiguration (DPR) model using a hypervisor to detect and recover a Multiprocessor System-On-Chip (MPSoC) architecture in case of error. Their method enhanced the system's flexibility but incurred reconfiguration delays.

Yang et al. [28] created an error bit prediction model of 3D Quad-Level Cell (QLC) NAND flash memory with high precision and statistical recovery of the error analysis. The biggest limitation was the extra hardware complexity of real-time predictions. In their study, Mohammed et al. [29] used the Hamming code and checksum to develop a merging technique circuit to help in error detection and correction that improved the fault tolerance of the digital circuit. The enhanced circuit complexity, however, caused an area overhead. Kim et al. [30] proposed 8-bit IMC approximate adders, which incorporated superior error mitigation measures that increased the reliability of arithmetic computations. The method had problems with accuracy in the compromise on the computation of the approximation.

3. Proposed TCAM-EC Architecture

This approach is not introduced within the current survey and makes the procedure of the traditional RAM-based error correction ineffective, as TCAM-EC in combination with ET-CLCS is utilized. The suggested TCAM-EC structure is presented in Figure 1. The proposed algorithm implements an HCRE technique that is efficient in encoding and decoding data stored in TCAM, unlike the conventional ECC and parity-based algorithms, which have a problem with high power usage and limited ability to correct errors.

A PSC unit is utilized to improve the error detection process, and this allows real-time error localization without extra memory overhead. To address this, an ACEM scheme is a dynamic scheme that adapts the level of redundancy depending on the severity of errors, to provide strong error mitigation without slowing down the high-speed TCAM operations. Integration of HCRE, PSC, and ACEM can be used to come up with a new error correction approach that is more efficient, faster, and effective than the former methods that would be used in high-density fault-resistant memory architectures. The stepwise operation mode suggested above can now be outlined in detail as follows:

Step 1: Input Data: The data that is to be stored must be considered.

Step 2: ET-CLCS with HCRE: The data security stage of the encoding step is performed before the transmission of the data using an HCRE technique. This method adds the concept of convolutional redundancy bits, which change according to the data pattern present, to make the structure self-adjustable and fault-tolerant. HCRE considerably reduces the total storage required, besides enhancing reliability by mitigating errors.

Step 3: TCAM-Based Data Storage and Read-Write Operations: The proposed algorithm includes data storage in TCAM instead of RAM, which has been considered traditional up to the present time. The TCAM's opposing characteristic of offering faster search pertinent to traditional CAM leads to research and writing errors due to differentiation in procedures, deterioration, and the effect of environmental noise. Unlike other conventional RAM error correction mechanisms, this method uses redundancy encoding based on the convolutional logic to address the errors that occur in TCAM storage.

Step 4: ET-CLCS for Error Detection and Correction: The ET-CLCS is utilized in the error detection and correction stage. ET-CLCS is intended to ensure redundant-based PSC and ACEM error-checking, which does not reduce the computational performance. To sum up, the last correction mechanism to ensure TCAM-based storage systems have a low-latency and high-reliability error correction.

- PSC for Error Correction: Conventional syndrome computations based on error control coding are replaced with the PSC unit through multi-bit parity, along with the novel syndrome matrix. Thus, it helps in the real-

time detection of single-bit and multiple-bit errors without incurring extra clock cycles, as it is efficient in high-speed TCAM memory.

- ACEM for Error Correction: The ACEM scheme is used for error correction depending on the grade of an error. By nature, ACEM is an accessible error magnitude and selects an appropriate error masking for correction, but in a way that creates as little redundancy as possible. Unlike the standard fixed configuration of ECC error correction mechanisms, the ACEM has some flexibility in power consumption for effective enhancement of memory durability.

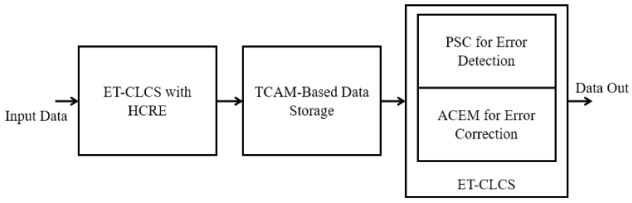


Fig. 1 Proposed TCAM-EC Architecture.

3.1. TCAM

The TCAM is a high-speed memory hardware that allows for the matching of the search data input against the stored entries in parallel. Different from the conventional memory that supports data access through specific addresses, TCAM can perform lookups across all the entries in parallel in a single clock cycle. This capability is quite useful in specific applications where decisions must be taken quickly and without much deliberation, as is evident in, for example, routing and forwarding of packets in a network. The architecture of the proposed TCAM system is depicted in Figure 2. Table 1 shows the algorithm procedure of the data writing operation in TCAM. The detailed data reading operations from TCAM are mentioned below in Table 2.

3.1.1. Write Operation

Writing data into TCAM is the procedural way of loading a key along with mask information, and in some implementations, priority information also. The data key is the data to be stored, while the mask shows which bits will be of future use to retrieve information since it opens "do not care" bits. To write data, the key, mask, and priority are written by the system into the TCAM. They then store this triplet in one of the memory arrays of the TCAM and, therefore, make it retrievable for other search operations. This process makes it possible for the TCAM to support complex matching since some predicates can specify which bits need to be used in the search.

3.1.2. Data Read Operation

There is no direct method of reading data from the TCAM since it is optimized for search operations and not for reading operations, as in the case of conventional memory systems. However, some TCAM architectures let the information stored in the entries be read by writing the address of that area in the memory array. In such a case, the system gives the physical address of the required index, and the TCAM gives the key, mask, and priority at that address. The update mechanism of this feature is also essential for controlling the entries stored in the TCAM and to guarantee that the memory has the most up-to-date information for search operations.

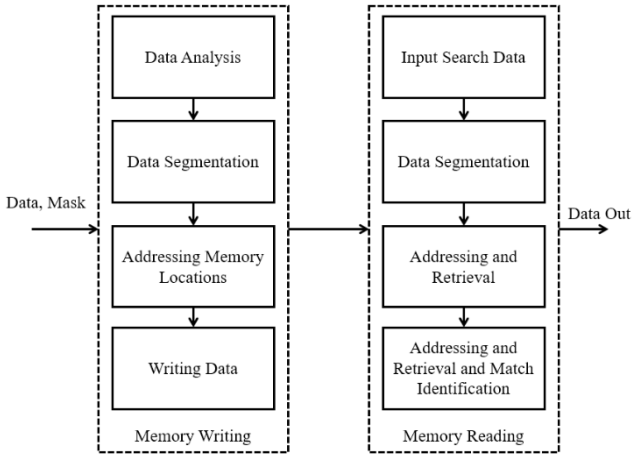


Fig. 2 Proposed TCAM Architecture.

Table 1. Data Write Operation in TCAM.

Input: Data, Mask **Output:** Data storage.
Step 1: Data Analysis: The data to be written, known as the key, is prepared alongside a corresponding mask. **Step 2:** The mask determines which bits in the key are significant ('1'), not significant ('0'), and "do not care" bits ('x'), allowing for flexible matching criteria. **Step 3: Data Segmentation:** The data is divided into multiple slices or chunks. Each slice corresponds to a segment of the TCAM's memory architecture. **Step 4: Addressing Memory Locations:** For each slice, the system calculates the specific memory addresses to write the data. If a slice contains "do not care" bits, the system must account for all possible combinations of these bits, leading to multiple addresses being targeted for that slice. **Step 5: Writing Data:** At each determined address, the system sets a bit corresponding to the entry number to '1', indicating the presence of the key at that location. This process is repeated for all slices, ensuring the entire key is accurately represented in the TCAM.

3.1.3. Search Operation

The unusual kind of operation of TCAM is a high-speed search operation that is the primary function of the device. In the searching process, the system offers a search key to the TCAM. Besides the search key a TCAM contains, all keys are compared at once, and the corresponding masks define the bits that matter in further comparison. Potential matches

are then potential entries that contain information matching the search key of the present system, according to the masks chosen by the user. In the case of having two or more entries matching the same pattern, the TCAM will use the priority values assigned to the entry as a basis for choosing the correct entry to use. The result matching the highest priority is taken, and the response connected with it is given as the result of the search. This parallel comparison mechanism enables TCAMs to perform search operations in one clock cycle, which proves helpful in improving certain operations, like a router's IP address lookup and access control list scan.

Table 2. Data Read (Search) Operation in TCAM.

Input: Stored data **Output:** Data out.
Step 1: Input Search Data: The system receives an input data (key) that needs to be searched within the TCAM. **Step 2: Data Segmentation:** Like the write operation, the input data (key) is divided into slices matching the TCAM's architecture. **Step 3: Addressing and Retrieval:** Each slice of the input key addresses a specific location in the TCAM. The data retrieved from these locations indicates which stored entries match the input slice. **Step 4: Aggregation of Results:** The system performs a logical AND operation across the results from all slices. This combined result yields a vector indicating which entries in the TCAM match the entire input key. **Step 5: Match Identification:** If multiple entries match, the TCAM identifies the highest-priority match based on predefined criteria, such as the lowest or highest address. The system then outputs the matching entry's identifier, facilitating rapid data retrieval.

3.2. ET-CLCS

The ET-CLCS is a newly proposed technique mainly for accelerating the error detection and correction of the TCAM-based storage system. The following is the proposed architecture of ET-CLCS, as shown in Figure 3. It incorporates a double-layer convolutional logic for redundancy-level management. HCRE is utilized to incorporate adaptive redundancy bits to make the codec exceptionally resistant to errors with minimal additional payload. Thus, to minimize the detection latency in the ET-CLCS, the PSC operates in parallel, in conjunction with the multi-bit parity and syndrome matrix schemes of the redundancy bit regions. Moreover, the Adaptive ACEM technique carries out correction methods among the data depending on the error percentage, allowing for making data redundancy and storage affordable at the same time. Here, the current generation of error correction mode ET-CLCS does not have fixed redundancy levels like most other methods for correcting errors, but offers dynamic redundancy patterns for enhanced computation, especially for high-speed memory.

This innovative scheme improves the error tolerance of TCAM-based memory systems without affecting computational capability. The convolutional logic works in parallel; the first layer is used for error detection, and the second layer is used for error correction, which reduces the response time and enhances capability. The HCRE, PSC, and ACEM, together within the ET-CLCS system, provide three-level protection to overcome single-bit and multi-bit errors, besides ensuring low-power and high-speed operation.

3.2.1. HCRE

To increase the absolute reliability and accuracy of the data stored in TCAM, the HCRE has been equipped with adaptively changeable redundancy bits. At the basis of HCRE, there is a generator matrix (G), and this matrix is constructed using the data patterns stored within a particular HCRE with the aim of optimizing for fault tolerance while at the same time ensuring minimal storage use. Compared to procedures for transmitting parity information applied in HCRE, which is based on syllables, it does not have a constant redundancy as in fixed redundancy. Instead, the generator matrix depends on the statistical characteristics of the data to be transmitted. It is possible to ensure that the encoded data is highly reliable without being optimally accompanied by high redundancy, especially when used for high-speed storage devices. The original and redundant bits of data are then written into the TCAM for fault tolerance in terms of read operations.

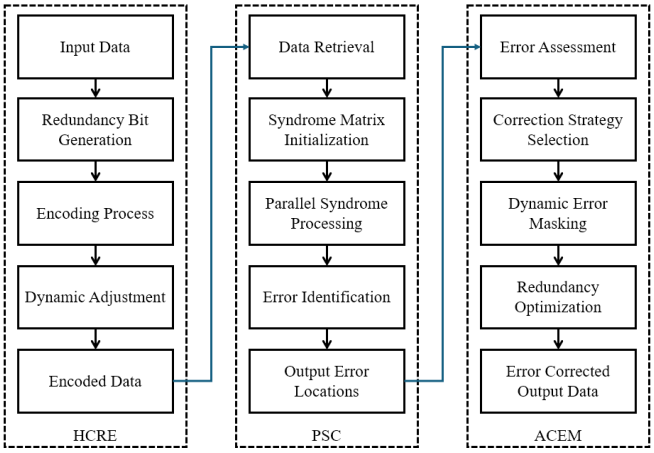


Fig. 3 Proposed ET-CLCS Architecture.

One of the major advantages of using HCRE is that it makes it possible to adapt the redundancy patterns and, consequently, the error tolerance of the stored information on the fly. Of these, the conventional approaches use static redundancy matrices, which may not be effective for dealing with varying error distributions. On the other hand, HCRE uses a dynamic redundancy matrix based on observed data to implement redundancy to subspaces that have a higher probability of error. Specifically, it helps to apply intelligent redundancy allocation within high-speed TCAM

architectures where error correction should be provided without significant delays. The HCRE incorporates redundancy into the stored data in a manner that allows the system to identify the flawed units and rectify them without having to work with costly computations as used in ECCs.

Table 3. HCRE Algorithm

Input: Input data **Output:** Encoded data.
Step 1: Input Data: Consider the input data as a vector d of length *k*. **Step 2: Redundancy Bit Generation:** Construct a generator matrix G that adapts based on the data patterns identified in d. This matrix combines the identity matrix I_{k} with a redundancy matrix P, forming $G = [I_{\mathrm{k}} \mid P]$. **Step 3: Encoding Process:** Compute the encoded data vector c by multiplying the data vector d with the generator matrix G: $c = d \times G$ (1) This operation integrates redundancy bits with the original data. **Step 4: Dynamic Adjustment:** Continuously monitor the data patterns in d and adjust the redundancy matrix P in real time to maintain optimal error resilience. **Step 5: Encoded data:** Store the encoded data vector c into TCAM.

3.2.2. PSC

At the right moment, the PSC algorithm is a fast error detection algorithm that is appropriate for a memory system using TCAM. Unlike the previous error-checking methods, such as serial processing, PSC uses the Parity-Check Matrix to perform the Syndrome calculations in parallel for each of the data bits. Based on the stored redundancy structure, the source is obtained to ensure that error detection is in synch with the encoded data. Thus, PSC effectively demonstrates whether errors exist by comparing the stored data with the parity-check matrix. It involves a systematic verification method that effectively minimizes the computational burden tied to the ECC-based error detection method.

Another advantage of PSC is its two-dimensional form, where it can compute more than one syndrome bit at once. In the older techniques of error detection, an individual syndrome computation is carried out for each bit, resulting in a latency in processing. However, PSCs process all stored bits simultaneously using a structured parity-check matrix. It further enables synchronous error detection, which is very effective in high-speed TCAM designs. No algebraic errors are to be determined in parallel to the memory operations; thus, optimization of latency is maintained at its highest level of efficiency.

The other feature of interest, particularly appreciated in PSC, is the capability to localize errors by computing the syndrome. In cases where the said syndrome vector is not equal to zero, the algorithm can not only detect the presence of the error but also the place of the error and fix it. It is especially useful in TCAM-based storage systems because frequently accessed and real-time integrity information retrieval is necessary. About the reduction of sequential error detection, PSC increases the computational performance by numerous folds, but, at the same time, makes the results rather accurate. So, it was a perfect approach for conventional memory architectures with low error rates and high throughput for processing in tandem with the memory module.

Table 4. PSC Algorithm.

Input: Errored Encoded Data **Output:** Error Locations.
Step 1: Data Retrieval: Access the stored or received encoded data vector r of length n for integrity verification. **Step 2: Syndrome Matrix Initialization:** Formulate the parity-check matrix H, structured as $H = [P^{\mathrm{T}} \mid I_{\mathrm{n-k}}]$, where P^{T} is the transpose of the redundancy matrix P. **Step 3: Parallel Syndrome Processing:** Compute the syndrome vector **s** by multiplying the parity-check matrix H with the transpose of the received vector r: $s = H \times r^{\mathrm{T}}$ (2) **Step 4: Error Identification:** Analyze the syndrome vector$s$. A zero vector indicates no errors, while a non-zero vector pinpoints the location and type of errors present. **Step 5: Output Error Locations:** Immediately flag detected errors for correction, ensuring minimal latency in high-speed memory applications.

3.2.3. ACEM

The ACEM algorithm is efficient and intended for the correction of error detection in high-operation TCAM-based storage systems. Conventional error correction, as we have seen, uses static redundancy, while for dynamic redundancy, the correction is done according to the error patterns detected. In the case of an error, it analyzes the gravity of the error and the extent to which the accuracy of the results has been affected, and such trivial errors are corrected with much less computation. This adaptability makes ACEM a very useful in-memory architectures that need real-time processing as it does not suffer from the problems of the other methods of error correction. Thus, the required corrections improve the reliability and lifetime of memory systems due to the high efficiency of ACEM.

Another interesting feature of ACEM is the ability to select the correction strategy in parallel to improve the performance of error masking in the case of multi-check faults. Unlike conventional error correction schemes, ACEM selects a correction vector depending on the syndrome computed at present. This kind of selection is parallelized so

that multiple locations with errors are processed at the same time, thus minimizing the amount of time taken to correct them. In conventional serial correction techniques, corrections are done one by one, which is time-consuming, adding delay in computational processes. However, ACEM can correct vector errors within T bits instantly by using the adaptive correction logic, which makes it highly suitable for high-speed TCAMs.

The second difference of ACEM is redundancy optimization, which ensures the overhead of the redundant codes that are deliberately used in the memory and the ability to correct resultant errors. ACEM does not fix the redundancy structure but adapts the parity-check matrix and redundancy bits to achieve the lowest storage overhead of redundancy bits while providing maximum error protection. The longstanding conventional ECC-related approaches have become rather wasteful in terms of providing extra capacity, which is not effectively utilized to enhance error correction effectiveness. ACEM overcomes this limitation by proactively allocating redundancy in accordance with actual error tendencies to avoid over-redundancy.

Table 5. ACEM Algorithm

Input: Error locations, Errored encoded data **Output:** Corrected data out.
Step 1: Error Assessment: Evaluate the syndrome vector*s* to determine the error's severity and impact on data integrity. **Step 2: Correction Strategy Selection:** Based on the error assessment, select an appropriate error correction vector **e** corresponding to the identified error pattern. **Step 3: Dynamic Error Masking:** Apply the error correction by adjusting the received vector r: $c' = r \oplus e$ (3) where $\bigoplus$ denotes bitwise addition modulo 2. **Step 4: Redundancy Optimization:** Adjust the redundancy matrix P parity-check matrix H to balance error correction capability with storage efficiency.

4. Results and Discussions

This section has determined the increase and decrease in the various methods for area, delay, and power introduced in this paper while comparing them under the same platform. All forms of implementations are now written using Verilog code and run on the Xilinx Vivado software to bring consistency. It also makes it possible to have the TCAM-based memory system performance analysis carried out accurately and in a systematic manner.

4.1. Simulation Results

The proposed TCAM-EC simulation is presented in Figure 4, which displays the read and write functions. The reset input is high during the first six ns, allowing all the registers and memory elements in the system to be initialized before the system enters normal operation. This is followed by the de-assertion of the reset (to zero) to make data transactions. The Clock (CLK) signal is running at a period of two ns, assuring the rapid data processing. The Mask signal (address signal) is the signal that enables the memory locations that are going to be used during read/write. The Din input values are 01 and 02, and they increase gradually, and the data being written into the memory is in sequence. The Enable (EN) signal is set to high throughout the simulation so that the memory module will stay operational. Write Enable (WE) is activated between 20 ns and 6ns, and during this window only, the data can be written into the memory.

The stored data is the written information that is at the given memory addresses. Write Enable (WE) is activated between 20 ns and 6ns, and during this window only, the data can be written into the memory. The stored data is the written information that is at the given memory addresses. The WE signal is then de-asserted (goes to zero), and the Read Enable (RD) signal is activated after 20 ns, and the read operation is triggered. This stage involves the retrieval of the stored data, and then the error detection and correction systems are applied to ensure the integrity of the data. The system manages to detect and rectify errors in the system and produce a zero-error output. This process confirms the strength of the proposed TCAM-EC design in supporting both the read and write with high accuracy.

Figure 5 is based on content searching, which is one of the major characteristics of the proposed TCAM-EC system. Then, the reset input is active during the initial six ns and is then on to permit normal operation. The clock signal is a two-ns periodic signal, which means that the execution of operations is synchronous. The start of Din input is 01 and 02, and it progressively goes up with all the stored data values. The signal in the EN signal is high during the simulation, and this ensures the memory is active. As compared to the read-write operation, in this case, the WE signal is not just triggered, but rather, constant data can be written in the memory.

In the meantime, the RD signal remains low, and the write and search operations are carried out. The Mask signal is configured to zero, and this implies that all the addresses stored in it can be searched in terms of content. At a point where the system identifies repeated data (i.e., similar keys recurrently occur in the system), the Match output is set with the value of active high, and this confirms that there were duplicates in the memory. Additionally, the repeated content's address is captured and generated as match_addr, allowing efficient retrieval and processing of matching entries. This simulation highlights the capability of the TCAM-EC to perform high-speed, efficient content searching while ensuring data integrity.

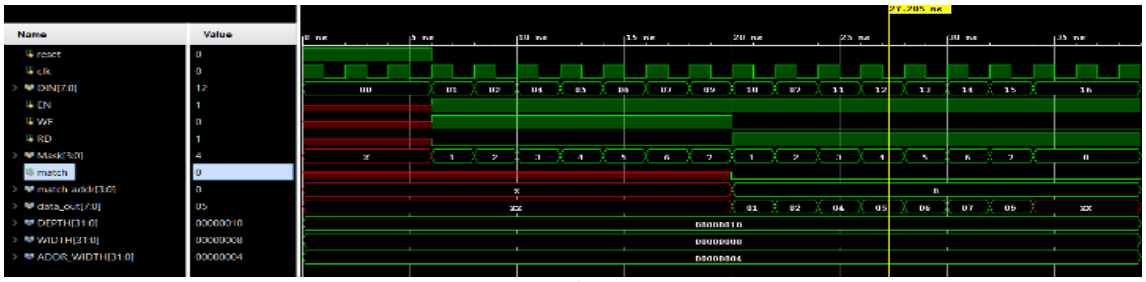


Fig. 4 Simulation Outcome for Read-Write Operations

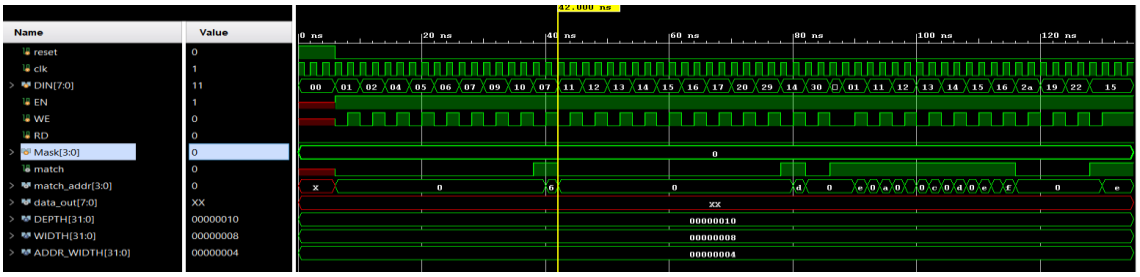


Fig. 5 Simulation Outcome for Content Searching Operations

The CAM-HD [17] and the proposed TCAM-EC architectures' area performance comparison are illustrated in Figure 6 in terms of LUTs, LUTRAM, and FF. From the above proposed TCAM-EC, it is evident that there is a considerable saving of resources in today's hardware systems. Further, 1665 LUT numbers in CAM-HD are significantly decreased to 838 in TCAM-EC, which is about a 49.7% reduction, and it claims that this architecture is more Area efficient. LUTRAM usage is constant at 8 for both architectures, which means that the modification does not change storage that is related to lookups. Also, the amount of FFs is reduced from 610 in CAM/HHD to 340 in TCAM/EC, which is a decline of 44.3%. This decrease in hardware utilization translates into lower complexity, reduced power consumption, and improved scalability for memory-intensive applications. The high area efficiency indicates the benefit of the TCAM-EC implementation to the traditional CAM-HD implementation.

Resource	Estimation	Available	Utilization %
LUT	1665	134600	1.24
LUTRAM	8	46200	0.02
FF	610	269200	0.23
IO	25	500	5.00
BUFG	1	32	3.13

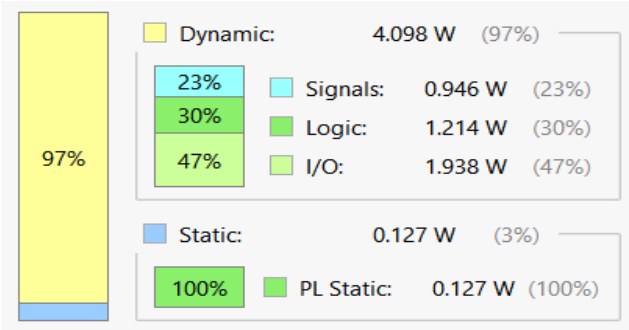
(a)

Resource	Estimation	Available	Utilization %
LUT	838	133800	0.63
LUTRAM	8	46200	0.02
FF	340	267600	0.13
IO	30	500	6.00
BUFG	1	32	3.13

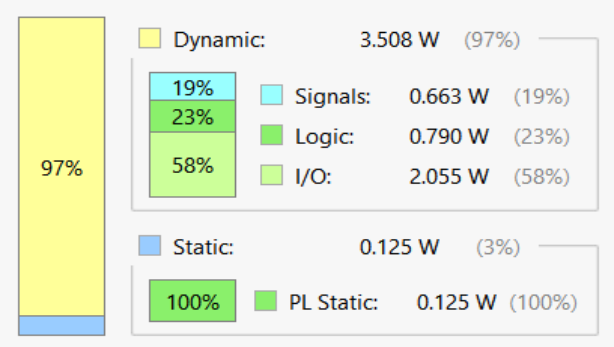
(b)

Fig. 6 Area Performance Analysis. (a) CAM-HD [17]. (b) Proposed TCAM-EC

Figure 7 shows the comparison of the power consumption of CAM-HD when compared with the proposed TCAM-EC, due to dynamic power, static power, and total power comparison. The signal, logic, and I/O components of the dynamic power are significantly lower in TCAM-EC. Compared to CAM-HD, which consumed 4.098 W in dynamic power consumption, the TCAM-EC consumes 3.508 W in dynamic power consumption, resulting in a 14.4% difference. The signal power, logic power, and I/O power all change slightly, but these changes are less when starting with 0.946 W to 0.663 W, then to 1.214 W to 1.938 W, and finally 2.055 W to 2.055 W, respectively. There exists a slight increase in the I/O power, but the total amount of power saved by reducing signal and logic power is tremendous. The power level at rest is virtually the same: 0.127 W with CAM-HD and 0.125 W with TCAM-EC. As a result, the overall power consumption reduces to 3.63 W in TCAM-EC as compared to 4.2 W in CAM-HD, with a difference of 13.6%. This enhancement implies that the TCAM-EC architecture is more power-saving without compromising stable functionality and is therefore a better solution to power-sensitive applications.



(a)



(b)

Fig. 7 Power Performance Analysis. (a) CAM-HD [17]. (b) Proposed TCAM-EC

Figure 8 presents the setup delay analysis of CAM-HD and TCAM-EC, which includes logic delay, net delay, and total setup time. The setup delay is one of the critical aspects determining the performance and timing closure of the circuit. Thus, the proposed TCAM-EC results in a smaller number of logic delays, net delays, and total setup delays than CAM-HD achieves. That is, the basic logic delay is brought down from 2.418 ns in CAM-HD to 2.388 ns in TCAM-EC; although this reduction is still confined to a minimal level, it denotes a certain improvement in the circuit design. Similarly, the net delay decreases and becomes more improved and optimized by decreasing from 9.524 ns in CAM-HD to 8.731 ns in TCAM-EC, which results in an 8.3% improvement. Altogether, the cumulative worth of these decreases comes to a total reduction of setup delay from 11.942ns in CAM-HD to 11.119ns in TCAM-EC, therefore a saving of 6.9%. The reduction of this setup delay makes it easier to operate high-speed applications while processing data from the memory.

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	8	5	65	T2/loc_reg[1]/C	T2/P_reg[3][1]/CE	11.942	2.418	9.524
Path 2	∞	8	5	65	T2/loc_reg[1]/C	T2/P_reg[3][2]/CE	11.942	2.418	9.524
Path 3	∞	8	5	65	T2/loc_reg[1]/C	T2/P_reg[3][6]/CE	11.942	2.418	9.524
Path 4	∞	8	5	65	T2/loc_reg[1]/C	T2/P_reg[3][0]/CE	11.709	2.418	9.291
Path 5	∞	8	5	65	T2/loc_reg[1]/C	T2/P_reg[3][3]/CE	11.709	2.418	9.291
Path 6	∞	8	5	65	T2/loc_reg[1]/C	T2/P_reg[3][4]/CE	11.709	2.418	9.291
Path 7	∞	8	5	65	T2/loc_reg[1]/C	T2/P_reg[3][5]/CE	11.709	2.418	9.291
Path 8	∞	8	5	65	T2/loc_reg[1]/C	T2/P_reg[3][7]/CE	11.698	2.418	9.280
Path 9	∞	8	5	65	T2/loc_reg[1]/C	T2/P_reg[9][5]/CE	11.669	2.418	9.251
Path 10	∞	8	5	65	T2/loc_reg[1]/C	T2/P_reg[11][2]/CE	11.534	2.413	9.121

(a)

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	8	5	79	T1/loc_reg[1]/C	T1/P_reg[12][3]/CE	11.119	2.388	8.731
Path 2	∞	8	5	79	T1/loc_reg[1]/C	T1/P_reg[12][7]/CE	11.119	2.388	8.731
Path 3	∞	8	5	80	T1/loc_reg[0]/C	T1/P_reg[4][3]/CE	11.095	2.417	8.678
Path 4	∞	8	5	80	T1/loc_reg[0]/C	T1/P_reg[4][6]/CE	11.095	2.417	8.678
Path 5	∞	8	5	79	T1/loc_reg[1]/C	T1/P_reg[1][3]/CE	11.085	2.390	8.695
Path 6	∞	8	5	79	T1/loc_reg[1]/C	T1/P_reg[1][5]/CE	11.085	2.390	8.695
Path 7	∞	8	5	79	T1/loc_reg[1]/C	T1/P_reg[1][6]/CE	11.085	2.390	8.695
Path 8	∞	8	5	80	T1/loc_reg[0]/C	T1/P_reg[4][0]/CE	11.053	2.417	8.636
Path 9	∞	8	5	80	T1/loc_reg[0]/C	T1/P_reg[4][1]/CE	11.053	2.417	8.636
Path 10	∞	8	5	80	T1/loc_reg[0]/C	T1/P_reg[4][2]/CE	11.053	2.417	8.636

(b)

Fig. 8 Setup Delay Analysis. (a) CAM-HD [17]. (b) Proposed TCAM-EC

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 11	∞	2	1	12	T1/P_reg[15][0]/C	T1/P_reg[15][1]/D	0.358	0.255	0.103
Path 12	∞	1	1	5	T1/count_reg[8]/C	T1/max_reg[8]/D	0.361	0.193	0.168
Path 13	∞	1	1	5	T1/count_reg[14]/C	T1/max_reg[14]/D	0.367	0.193	0.174
Path 14	∞	1	1	5	T1/count_reg[13]/C	T1/max_reg[13]/D	0.368	0.193	0.175
Path 15	∞	1	1	5	T1/count_reg[15]/C	T1/max_reg[15]/D	0.378	0.193	0.185
Path 16	∞	2	1	6	T1/State_reg[0]/C	T1/full_reg/D	0.384	0.255	0.129
Path 17	∞	1	1	5	T1/count_reg[18]/C	T1/max_reg[18]/D	0.407	0.193	0.214
Path 18	∞	2	1	11	T1/P_reg[3][3]/C	T1/P_reg[3][3]/D	0.409	0.255	0.154
Path 19	∞	2	1	10	T1/P_reg[11][4]/C	T1/P_reg[11][4]/D	0.417	0.255	0.162
Path 20	∞	1	1	5	T1/count_reg[11]/C	T1/max_reg[11]/D	0.417	0.193	0.224

(a)

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 11	∞	1	1	10	T1/count_reg[5]/C	T1/max_reg[5]/D	0.374	0.193	0.181
Path 12	∞	1	1	10	T1/count_reg[6]/C	T1/max_reg[6]/D	0.380	0.193	0.187
Path 13	∞	1	1	49	T1/count_reg[3]/C	T1/max_reg[3]/D	0.392	0.193	0.199
Path 14	∞	1	1	10	T1/count_reg[4]/C	T1/max_reg[4]/D	0.392	0.193	0.199
Path 15	∞	2	1	10	T1/P_reg[8][4]/C	T1/P_reg[8][4]/D	0.403	0.255	0.148
Path 16	∞	2	1	9	T2/P_reg[6][4]/C	T2/P_reg[6][4]/D	0.407	0.255	0.152
Path 17	∞	1	1	20	U1/top1/wptr_reg[2]/C	U1/top3/da...AMA/WADR2	0.409	0.177	0.232
Path 18	∞	1	1	20	U1/top1/wptr_reg[2]/C	U1/top3/da...A_D1/WADR2	0.409	0.177	0.232
Path 19	∞	1	1	20	U1/top1/wptr_reg[2]/C	U1/top3/da...AMB/WADR2	0.409	0.177	0.232
Path 20	∞	1	1	20	U1/top1/wptr_reg[2]/C	U1/top3/da...B_D1/WADR2	0.409	0.177	0.232

(b)

Fig. 9 Hold Delay Analysis. (a) CAM-HD [17]. (b) Proposed TCAM-EC.

As illustrated in Figure 9, the hold delay is a measure specifying the logic delay, net delay, and the total hold time of CAM-HD and the proposed TCAM-EC. The hold delay is an important parameter in the preservation and maintenance of data stability, and in the absence of timing violations. The logic delay decreases by 0.193-0.177ns, which represents an 8.3% decrease in the logic delay of CAM-HD and TCAM-EC, respectively. There is, however, a slight increment of the net delay between CAM-HD (0.224 ns) and TCAM-EC (0.232 ns), which represents a slight trade-off. Although this represents a minor increase in net delay, the overall hold delay goes down in the CAM-HD; this is 0.417 ns, compared to TCAM-EC, which is 0.409 ns. The hold delay is reduced, which means that it is more stable and less vulnerable to timing errors, which once again supports the validity of the offered TCAM-EC design as suitable in the high-performance environment.

4.2. Objective Comparison

The area performance of the various memory architectures, such as ETCIM [11], RaPC [23], CAM-HD [17], and the proposed TCAM-EC, is compared using 3 main parameters: LUTs, LUTRAMs, and FFs, as shown in Table 6. The proposed TCAM-EC suggests a considerable decrease in the area consumption as compared to the other designs. The utilization of the LUT used in ETCIM [11] is 2103, that of RaPC [23] is 1872, and that of CAM-HD [17] is 1665, yet the suggested TCAM-EC uses 838, which is approximately a 49.7 percent decrease over CAM-HD [17]. This decrease is vital because LUTs directly affect the efficiency of resources consumed by FPGAs and the scalability of the design. The LUTRA usage is fair in all architectures. ETCIM [11] implements 18 LUTRAMs, 13 of which were needed in RaPC [23], and only 8 LUTRAMs in CAM-HD [17] and TCAM-EC. The low percentage of difference in the usage of LUTRAM indicates that the TCAM-EC does not play a big role in LUTRAM usage, but improves the usage of other architectural elements. The FF utilization, which shows the number of used sequential elements, has a similar trend. ETCIM [11] uses 1028 FFs, RaPC [23] uses 872, and CAM-HD [17] uses 610. The proposed TCAM-EC also decreases the use of FF to nearly 340; this is about a 44.3 percent cutdown of what CAM-HD demands [17].

Table 6. Area Performance Comparison of Various Memories.

Parameter	ETCIM [11]	RaPC [23]	CAM-HD [17]	Proposed TCAM-EC
LUT	2103	1872	1665	838
LUTRAM	18	13	8	8
FFs	1028	872	610	340

Table 7 shows the power performance comparison of various memory architectures, including ETCIM [11], RaPC [23], CAM-HD [17], and the proposed TCAM-EC, which is analyzed based on dynamic power, signal power, logic power, I/O power, static power, and total power consumption. The proposed TCAM-EC also shows that it has less dynamic power and total power consumption compared with the previous architectures. The dynamic power consumption of the ETCIM is 6.201 W, the RaPC is 5.696 W, and the CAM-HD is 4.098 W. The following TCAM-EC cuts down dynamic power to 3.508 W, 14.4% less than the CAM-HD [17]. It is due to a reduction in the logic level of optimization and a low level of switching activity. Likewise, the signal power consumption demonstrates the same kind of trend where ETCIM [11] and RaPC [23] exert a power of 1.500 W and 1.320 W, respectively, whereas CAM-HD [17] demands a mere 0.946 W. The proposed TCAM-EC reduces it to 0.663 W, which is 29.9% better compared to CAM-HD [17] to increase energy efficiency at large. The logic power consumption, which calculates the power to compute, is equal to 1.820 W for ETCIM [11], 1.600 W for RaPC [23], and 1.214 W for CAM-HD [17]. It should be noted that the proposed TCAM-EC even further reduces it to 0.790 W, which is assessed to be 34.9% down from CAM-HD [17], which, in terms of computation, has been efficiently designed.

The I/O power consumption presents an interesting observation where ETCIM [11] uses 2.420 W, RaPC [23]

consumes 2.310 W, and CAM-HD [17] consumes 1.938 W. However, the TCAM-EC slightly increases it to 2.055 W, which is a 6% increase compared to CAM-HD [17]. It is due to enhanced interconnectivity or increased communication efficiency within the design. The static power consumption remains almost constant across all architectures, with ETCIM [11] and RaPC [23] at 0.127 W, CAM-HD [17] also at 0.127 W, and TCAM-EC slightly reducing it to 0.125 W. Finally, the total power consumption is a crucial performance metric. ETCIM [11] consumes 6.328 W, RaPC [23] consumes 5.823 W, CAM-HD [17] consumes 4.2 W, and the proposed TCAM-EC achieves the lowest total power consumption of 3.63 W. This represents a 13.6% power efficiency improvement over CAM-HD [17], making TCAM-EC the most energy-efficient design among the evaluated architectures.

Table 7. Power Performance Comparison of Various Memories.

Power Parameter	ETCIM [11]	RaPC [23]	CAM-HD [17]	Proposed TCAM-EC
Dynamic Power (W)	6.201	5.696	4.098	3.508
Signal Power (W)	1.500	1.320	0.946	0.663
Logic Power (W)	1.820	1.600	1.214	0.790
I/O Power (W)	2.420	2.310	1.938	2.055
Static Power (W)	0.127	0.127	0.127	0.125
Total Power (W)	6.3283	5.823	4.2	3.63

Table 8. Set up a Delay Performance Comparison of Various Memories.

Delay Parameter	ETCIM [11]	RaPC [23]	CAM-HD [17]	Proposed TCAM-EC
Logic Delay (ns)	4.010	3.008	2.418	2.388
Net Delay (ns)	16.009	12.930	9.524	8.731
Total Setup Delay (ns)	20.019	15.938	11.942	11.119

Table 9. Hold Delay Performance Comparison of Various Memories.

Delay Parameter	ETCIM [11]	RaPC [23]	CAM-HD [17]	Proposed TCAM-EC
Logic Delay (ns)	0.301	0.252	0.193	0.177
Net Delay (ns)	0.323	0.281	0.224	0.232
Total Hold Delay (ns)	0.623	0.513	0.417	0.409

Table 8 also proves the setup delay performance analysis of the different memory architectures, including ETCIM [11], RaPC [23], CAM-HD [17], and the proposed TCAM-EC by evaluating logic delay, net delay, and the total setup delay. Thus, the proposed TCAM-EC proposes an efficient architecture with better features that help in reducing the total setup delay, the logic and net delays for increased speed of operation of memory systems. The logic delay, which signifies the delay pertinent to the logic components inside, is 4.010 ns for ETCIM [11], 3.008 ns for RaPC [23], as well as 2.418 ns for CAM-HD [17]. The proposed TCAM-EC continues to provide an even lower logic delay of 2.388 ns, thus being 1.2% better than CAM-HD [17]. It means that there is outlining of newspaper advertising and minimizing of the used logic gates, as well as structuring of the circuit. The net delay, the delay that is, however, incurred due to the time taken to traverse through the interconnections, is lower in the proposed TCAM-EC architecture. Another work, namely ETCIM [11], takes a net delay of 16.009 ns, RaPC [23] has one of 12.930 ns, and CAM-HD [17] has one of 9.524 ns. The proposed CAM-EC reduces it to 8.731 ns by making it 8.3% less than the CAM-HD mentioned in [17]. This improvement emanates from improved routing and minimization of the wire delay, implying that signals will take the shortest time to be transmitted. Total setup delay, the net of logic delay, is another performance parameter that was explained above. Here, it is revealed that ETCIM [11] has the largest total setup delay, which is 20.019 ns, while RaPC [23] has 15.938 ns. Spatz tested CAM-HD [17] and reduced it to 11.942 ns; also, the proposed TCAM-EC attained the lowest total capabilities of setup delay at 11.119 ns. The improvement of 6.9% over CAM-HD [17], in general, increases the speed and frequency of memory motions.

In Table 9, the hold delay performance of various memory architectures, ETCIM [11], RaPC [23], CAM-HD [17], and TCAM-EC, is analyzed in terms of logic delay, net delay, and total hold delay. This paper puts forward how the TCAM-EC outdoes the other architectures in that it minimizes hold delays, enhances the stability of system timing, and negates other undesired timing fluctuations. The frequency-dependent delay and the frequency-independent delay are 0.196 ns for ETCIM [11], 0.201 ns for RaPC [23], and 0.145 ns for CAM-HD [17]. The further proposed TCAM-EC reduces it to 0.177 ns without any intermediate pass, which is 8.2% less than the CAM-HD [17]. This

reduction is realized using optimized circuit design and effective running of the gates. They also have the net delay that counts for the propagation delay of signals between the interconnected components, and these also show enhancements.

In terms of net delay, ETCIM [11] captures 0.323 of ns, RaPC [23] has 0.281, and CAM-HD [17] has 0.224 of ns. As for the proposed TCAM-EC, it implements a net delay slightly greater than 0.232 ns for better signal integrity, with greater synchronization in terms of timing. Therefore, the total hold delay, which is obtained by adding the logic and net delay, is 0.623 ns for ETCIM [11], 0.513 ns for RaPC [23], and 0.417 ns for CAM-HD [17]. The proposed TCAM-EC is even further reduced to 0.409 ns, a 1.9% improvement over the CAM-HD [17]. The reduction of access times also helps to reduce various timing problems that are likely to cause instabilities in memory functioning.

5. Conclusion

When it is combined with the ET-CLCS, TCAM-EC is a new and innovative approach presented to enhance memory dependability, speed, and durability in high-speed computing applications. Thus, by using the discussed method that involves PSC for real-time error checking and ACEM for dynamic redundancy, we can avoid problems that were associated with traditional RAM-based ECC and parity methods.

The Integration of these techniques provides high speed, low power, and built-in testing for TCAM applications, which makes it suitable for future memory designs. The future work can target improving the factors of convolutional redundancy encoding for improving the efficiency of the multi-bit error correction; developing machine learning methods for incipient error prediction for any faults to arise; expanding the work's application to other memory technologies like RRAM and MRAM for more applications in high-performance computing, AI accelerators, and cloud storage.

However, for further assessment in terms of real area, time, and frequency reduction in practice, as well as to validate the proposed circuits in an FPGA, further hardware implementation is a possibility.

References

[1] Tergel Molom-Ochir et al., "Advancements in Content-Addressable Memory (CAM) Circuits: State-of-the-Art, Applications, and Future Directions in the AI Domain," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 72, no. 8, pp. 3971-3982, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[2] Arsalan Dabbagh et al., "CD-MAC: Mixed-Signal Binary/Ternary In-memory Computing Accelerator for Power-constrained MAC Processing," *The Journal of Supercomputing*, vol. 81, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[3] Jeongmin Shin et al., "C2IM-NN: A Low-Power 3D Point Clouds Matching Processor with 1D-CNN Prediction and CAM-Based In-Memory k-NN Searching," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 72, no. 7, pp. 3286-3297, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[4] Chandramouli Amarnath et al., "Error Resilient Online Reinforcement Learning Using Adaptive Statistical Checks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 8, pp. 3112-3125, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[5] Maitreyi Ashok et al., "Digital In-Memory Compute for Machine Learning Applications with Input and Model Security," *IEEE Journal of Solid-State Circuits*, vol. 60, no. 9, pp. 3390-3402, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[6] Zahra Shirmohammadi, and Masoumeh Taali, "JoBiS: Joint Capacitance and Inductance Bit Stuffing CAC for Interposer based Multi-chip Deep Learning Accelerator," *Integration*, vol. 102, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[7] Swathi Mummadi et al., *Error Correction Methods for Protecting Quantum Information*, The Quantum AI Era of Neuromarketing, IGI Global, pp. 113-132, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[8] Nazim Altar Koca et al., "Exploring Error Correction Circuits on RISC-V based Systems for Space Applications," *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*, Singapore, Singapore, pp. 1-5, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[9] Luca Parrini et al., "Error Detection and Correction Codes for Safe In-Memory Computations," *2024 IEEE European Test Symposium (ETS)*, The Hague, Netherlands, pp. 1-4, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[10] Ting-An Lin, and Po-Tsang Huang, "Design Exploration of In-Situ Error Correction for Multi-Bit Computation-in-Memory Circuits," *2024 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Taipei, Taiwan, pp. 174-178, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[11] Yiqi Wang et al., "ETCIM: An Error-Tolerant Digital-CIM Processor with Redundancy-Free Repair and Run-Time MAC and Cell Error Correction," *2024 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, Honolulu, HI, USA, pp. 1-2, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[12] Hui Jin et al., "Novel Latin Square Matrix Code of Large Burst Error Correction for MRAM Applications," *Microelectronics Reliability*, vol. 162, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[13] Yichuan Bai et al., "A Compilation Framework for SRAM Computing-in-Memory Systems with Optimized Weight Mapping and Error Correction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 43, no. 8, pp. 2379-2392, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[14] Youngki Moon et al., "A Novel Prediction-Based Two-Tiered ECC for Mitigating SWD Errors in HBM," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 33, no. 2, pp. 488-498, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[15] Yogendra Gupta et al., *STT-MRAM based Cache Memory Design for Multicore Systems*, Circuit Design for Modern Applications, 1st ed., CRC Press, pp. 1-18, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[16] Suheib Alhiyari, Siti Hafizah AB Hamid, and Nur Nasuha Daud, "A Decentralized and TCAM-Aware Failure Recovery Model in Software Defined Data Center Networks," *Computers, Materials & Continua*, vol. 82, no. 1, pp. 1087-1107, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[17] Esteban Garzón et al., "A 128-kbit Approximate Search-Capable Content-Addressable Memory (CAM) with Tunable Hamming Distance," *IEEE Journal of Solid-State Circuits*, vol. 60, no. 8, pp. 3009-3019, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[18] Vani Marani et al., "Approach Toward In-Memory Computing-Based Hamming Code Implementation for Error Correction and Detection," *VLSI for Embedded Intelligence: Proceedings of the 27th International Symposium*, pp. 357-368, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[19] Yingjie Yu et al., "3D Self-Rectifying Memristive Ternary Content Addressable Memory for Massive and Exact In-memory Search," *Science China Information Sciences*, vol. 68, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[20] Aswin Sreekumar, Bolupadra Sai Shankar, and B. Naresh Kumar Reddy, "Integrating Error Correction and Detection Techniques in RISC-V Processor Microarchitecture for Enhanced Reliability," *Integration*, vol. 100, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[21] Seongyoon Kang, Chaehyeon Shin, and Jongsun Park, "Fault Bounding On-Die BCH Codes for Improving Reliability of System ECC," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 33, no. 5, pp. 1482-1486, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[22] Surendra Hemaram et al., "Asymmetric and Adaptive Error Correction in STT-MRAM," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 9, pp. 3336-3349, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[23] Ruifeng Tu et al., "RaPC: Raw Bit Error Rate Aware Polar Coding for 3-D nand Flash Memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 44, no. 9, pp. 3546-3559, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[24] Jaeho Shin, and Jungrae Kim, "ROSE: Reliability-Optimized OD-ECC and S-ECC Enhancements for HBM3," *2025 International Conference on Electronics, Information, and Communication (ICEIC)*, Osaka, Japan, pp. 1-4, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[25] Nilesh Kumar Jha, Huayan Guo, and Vincent K. N. Lau, "Robust In-Memory Computation with Bayesian Analog Error Mitigating Codes," *IEEE Transactions on Signal Processing*, vol. 73, pp. 534-548, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[26] Abdul Aziz et al., "Multi-bit Error Detection and Correction Technique using HVDK (Horizontal-Vertical-Diagonal-Knight) Parity," *Integration*, vol. 100, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[27] Jorge Cano-Páez et al., "Architecture for Error Detection and Recovery in MPSoCs: A Hypervisor Approach Using Dynamic Partial Reconfiguration," *IEEE Transactions on Nuclear Science*, vol. 72, no. 8, pp. 2636-3644. [CrossRef] [Google Scholar] [Publisher Link]

[28] Guangkuo Yang et al., "High-Precision Error Bit Prediction for 3D QLC NAND Flash Memory: Observations, Analysis, and Modeling," *IEEE Transactions on Computers*, vol. 74, no. 4, pp. 1392-1404, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[29] Mohammed Sami Mohammed et al., "Design a Merging Technique Circuit to Error Detection and Correction Based on Hamming Code and Checksum Using VHDL," *International Journal of Electrical and Electronics Research*, vol. 12, no. 4, pp. 1449-1460, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[30] Jinhyun Kim, and Youngmin Kim, "Novel 8-bit IMC Approximate Adders with Advanced Error Mitigation Techniques," *2025 International Conference on Electronics, Information, and Communication (ICEIC)*, Osaka, Japan, pp. 1-4, 2025. [CrossRef] [Google Scholar] [Publisher Link]